

```

1 /*****
2  * Program: DoubleDim.c
3  * Author: Sanjay Vyas
4  *
5  * Description
6  * This program shows how to create double dimension arrays and
7  * allocate memory for each type.
8  *****/
9
10 #include <stdio.h>
11 #include <stdlib.h>
12
13 #define R (3)
14 #define C (3)
15
16 int main()
17 {
18     int aa[R][C];    // Array of array
19     int *ap[R];      // Array of pointers
20     int (*pa)[C];    // Pointer to array
21     int **pp;        // Pointer to pointers
22
23     int i;           // Loop variable for first dimension
24     int j;           // Loop variable for second dimension
25
26     int val;         // User input
27
28     // 1. aa - Array of array required no allocation
29
30     // 2. ap - Array of pointers need to allocate columns
31     for (i = 0; i < R; i++)
32     {
33         ap[i] = (int *)malloc(C * sizeof(int));
34         if (NULL == ap[i])
35         {
36             fprintf(stderr, "Sorry, not enough memory\n");
37             return 1;
38         }
39     }
40
41     // 3. pa - Pointer to array needs to allocate rows
42     pa = (int (*)[C])malloc(R * sizeof(int[C]));
43     if (NULL == pa)
44     {
45         fprintf(stderr, "Sorry, not enough memory\n");
46         return 1;
47     }
48
49     // 4. pp - Pointer to pointer requires allocation for rows and columns
50     pp = (int**)malloc(R * sizeof(int*));
51     if (NULL == pp)
52     {
53         fprintf(stderr, "Sorry, not enough memory\n");
54         return 1;
55     }
56     for (i = 0; i < R; i++)
57     {
58         pp[i] = (int *)malloc(C * sizeof(int));
59         if (NULL == pp[i]) {
60             fprintf(stderr, "Sorry, not enough memory\n");
61             return 1;
62         }
63     }
64
65     // Read values from the user and print all 4 double dimension
66     for (i = 0; i < R; i++)
67     {
68         for (j = 0; j < C; j++)
69         {
70             printf("Enter value for Element[%d][%d]: ", i, j);
71             scanf("%d", &val);
72
73             // Assign the same value to all 4 double dim
74             aa[i][j] = ap[i][j] = pa[i][j] = pp[i][j] = val;
75         }
76     }
77
78     // Print array of array
79     printf("Printing array of array\n");
80     for (i = 0; i < R; i++)
81     {
82         for (j = 0; j < C; j++)
83             printf("%d\t", aa[i][j]);
84         printf("\n");
85     }
86     printf("\n");
87
88     // Print array of pointers
89     printf("Printing array of pointers\n");
90     for (i = 0; i < R; i++)
91     {
92         for (j = 0; j < C; j++)
93             printf("%d\t", ap[i][j]);
94         printf("\n");
95     }
96     printf("\n");
97
98     // Print pointer of array
99     printf("Printing pointer to array\n");
100    for (i = 0; i < R; i++)
101    {
102        for (j = 0; j < C; j++)
103            printf("%d\t", pa[i][j]);
104        printf("\n");

```

```
105 }
106 printf("\n");
107
108 // Print pointer of pointer
109 printf("Printing pointer to pointer\n");
110 for (i = 0; i < R; i++)
111 {
112     for (j = 0; j < C; j++)
113         printf("%d\t", pp[i][j]);
114     printf("\n");
115 }
116 printf("\n");
117
118
119 // Free up memory for ap, pa and pp
120 // 1. aa does not require free as it required no malloc
121
122
123 // 2. Free up for ap.. it requires a loop
124 for (i = 0; i < R; i++)
125     free(ap[i]);
126
127 // 3. Free up for pa.. it requires a single free
128 free(pa);
129
130 // 4. Free up pp.. it requires a loop and then one free for top level
131 for (i = 0; i < R; i++)
132     free(pp[i]);
133
134 free(pp);
135
136 return 0;
137 }
138
```